Scene Complexity: A measure for real-time stable haptic applications

Eric Acosta, Bharti Temkin
Department of Computer Science, Texas Tech University
Bharti.Temkin@coe.ttu.edu

Abstract

We discuss real-time issues of scene-complexity in order to frame a technical envelope for stable haptic applications. How large a scene can a haptic application support? Specifically, we try to determine the largest stable haptic scene possible when GHOST is used to develop an application. The scene consists of a number of non-overlapping primitives or polymesh objects. The scene complexity is measured as a number of primitive objects or polygons in polymesh objects that allow stable haptic interactions.

The initial data collected indicates that earlier versions of GHOST allow for more complex scenes. Each later version seems to reduce the number of objects that can be included in a scene. For example, Version 1.2 allows for inclusion of 1100 non-overlapping spheres in a stable haptic application. Version 2.0 allows only 770 (almost 30% less) such objects, while version 3.0 allows only 600 (almost 45% less). The system used for data collection is a Pentium III 500 MHz computer with 256 MB of RAM. This machine has an nVidia Riva TNT2 Ultra video card with 32 MB of memory and runs Windows NT Workstation 4.0 with Service Pack 6.0.

In order to quantify and understand these observations, we have developed an application that estimates the distribution of resources used within the haptic loop; the fractions used by collision detection, graphics, and haptic processes. This study provides a method for performance analysis of haptic systems. Scene complexity plays a key role in the creation of haptic applications by providing critical data needed to estimate the feasibility and performance limits for generic haptic environments. It thus should have a broad impact on the design of haptic applications.

1. Introduction

Scene complexity is a measurement on how complex a scene can get and still allow for stable haptic interactions. The real-time performance is typically reduced in proportion to the increase in the complexity of the scene. In fact, it is well known that the time to compute haptic interactions increases with the number of polygons. The approach taken in previous work was to make the haptic servo loop rate essentially independent of the number of polygons [1]. However, this invariance is obtained only after the contact is made with an object and while the object is being touched in the neighborhood of the proxy, while the proxy remains inside the object, making this an efficient haptic rendering technique for a single polymesh object. In this paper, the scene complexity is measured as a number of primitive objects or polygons in polymesh objects. By understanding the limits imposed by the complexity of the scene, we can estimate the feasibility and performance limitations of generic haptic environments.

First, we consider the greatest lower bound (GLB), the highest scene complexity for which the application always runs with stability and no errors. Next we consider the least upper bound (LUB), where some instability and errors are allowed to occur. When the number of objects is between GLB and LUB the application runs some of the times and produces errors or instabilities at other times. When the complexity exceeds LUB of the scene, the application produces errors instantly after haptics is initialized. This establishes a scene complexity stability range for haptic applications, as shown in Figure 1.

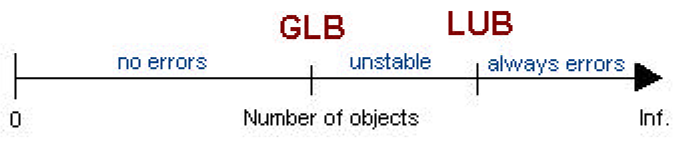

Figure 1: Scene complexity stability range.

2. Primitive Objects: Spheres and Box complexity tests

To establish the scene complexity, our application allows a tester to specify the number of primitive objects (in the x, y, and z directions) in the scene. The application automatically creates and spaces the objects to prevent them from overlapping as shown in Figure 2. The use of non-overlapping objects eliminates other factors affecting haptic load (hload) [2]. Hload is the time required to complete a haptic loop. Our application records the hload data with 10^{-3} ms precision and stores it in a file. In order to estimate the time distribution of the graphics, haptic, and collision detection tasks [3] within the 1 ms haptic loop, hload can be measured in several modes: touching (T) or not touching (NT) an object, graphics on (G) or off (NG), and removing geometry from the haptic scene graph (NH). By removing the geometry branch "hapticScene", as seen in Figure 3, the geometry can be eliminated from the haptic scene graph. The NH mode represents the time required to perform other duties of the haptic loop, such as the device position query and the scene graph traversal, without having to perform any collision detection on the geometry objects themselves.

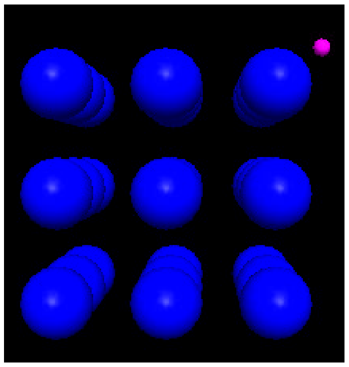
Figure 2: Spheres in a 3x3x3 (x,y,z) fashion.

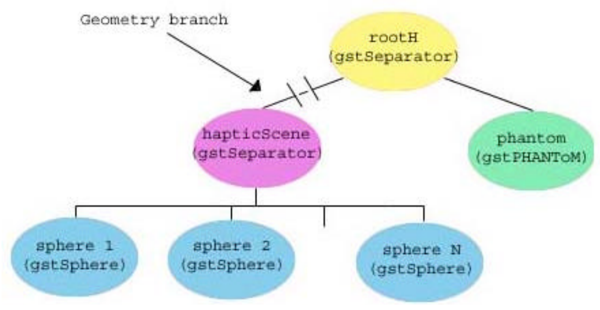

Figure 3: Sample scene graph representation.

The test data was collected five times for a combination of modes between G or NG, T or NT, and NH (G_T, G_NT, NG_T, NG_NT, G_NH, NG_NH). Starting with 8 (2x2x2) objects, the number of objects is increased by one in each direction (e.g. next test has 9(3x3x3) objects), until an error is generated for exceeding the 1 ms time constraint of the haptic duty cycle. At this stage, the number of objects is slowly decreased to identify the GLB. Next, the number of objects is increased to find the LUB, a point at which the application instantly produces an error.

Tests were conducted using three different versions of GHOST (1.2, 2.0, and 3.0) to evaluate performance changes from version to version. The GLBs and LUBs for the different versions are summarized in Table 1. The GLB between version 1.2 and 2.0 is reduced by about 30% and reduced by another 22% between versions 2.0 and 3.0 for spheres. Test results for version 3.0 are displayed in Figure 4. From the tests, we calculated that on average displaying graphics increased the hload by about 2-5% and touching an object increased it by about 7-18%. Our test results indicate that earlier versions of GHOST allowed for more complex scenes. Each subsequent version reduced the number of objects that could be included in the scene for a stable haptic application. The GLB between versions 1.2 and 2.0 is reduced by about 27% and is further reduced by 25% between versions 2.0 and 3.0 for boxes. In general our test show that hload increases

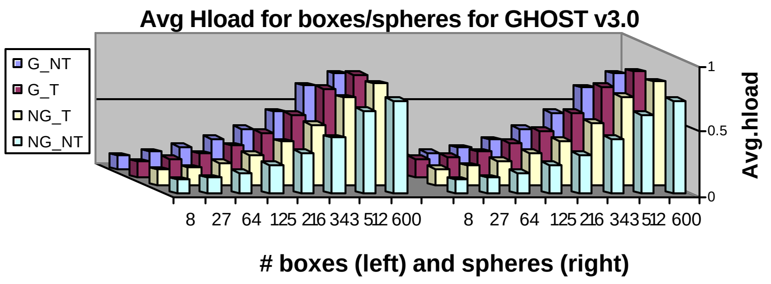

Figure 4: Avg Hload for GHOST v3.0

Version	Object	GLB	LUB
1.2	Box	1100	1188
	Sphere	1100	1200
2.0	Box	800	847
	Sphere	770	847
3.0	Box	600	729
	Sphere	600	729

Table 1: GLB and LUB for GHOST versions

fairly linearly as objects are added to the scene graph for every version of GHOST. Clearly, overhead was introduced from version to version. In case of no graphics and no haptic scene (NGNH), the average hloads were .05ms, .08ms, and .09ms for versions 1.2, 2.0, and 3.0 respectively. The data represents the time required to perform basic duties of the haptic loop, such as device position query and scene graph traversal for nodes other than geometry.

The hload percentage for collision detection is about 82-83% for spheres and 89-90% for boxes, for graphics it is about 3% for both spheres and boxes, and for touching an object it is about 11-13% for spheres and 6-8% for boxes.

3. Polymesh objects complexity tests

For a single polymesh object, a box made up of length, width, and height segments was used. These segments define the resolution of the object and determine the number of polygons that form it. The number of vertices (*nv*) and faces (*nf*) can be calculated as follows, where *l*, *w*, and *h* are the number of length, width, and height segments respectively:

$$nv = 2?\ [(l+1)(h+1) + (w-1)(h+1) + (w-1)(l-1)]$$

$$nf = 4?[(l?h) + (w?h) + (w?l)]$$

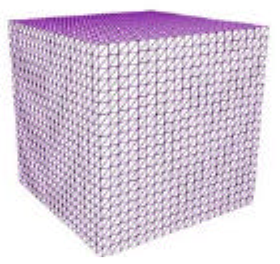

Figure 5: Polymesh box

Figure 5 is a wire frame rendering of the box showing how the object is made of triangular polygons. Starting with a 10x10x10 segment (602 vertices, 1200 polygons) box, the numbers of segments (length, width, and height) are increased by 10 for each test.

Figure 6 displays the average results of five test runs with GHOST version 3.0. According to these results, the hload varied by very little when the polygon count increased for a single object. However, when polygon counts exceeded about 120k, GHOST started to produce random errors, even though hload was under 0.2ms. From these results, for a single object with no overlapping polygons, we found the GLB and LUB to be 120k and 235k polygons respectively.

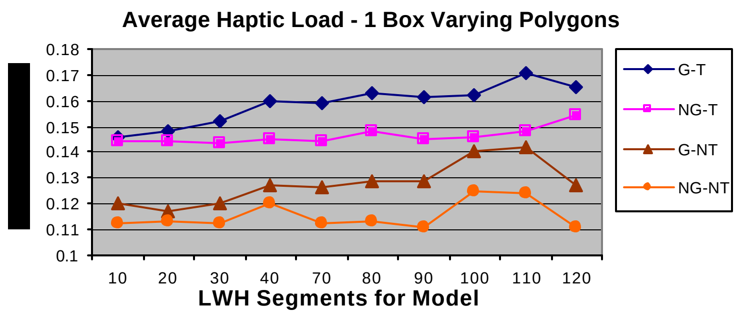

Figure 6: Avg test results for increasing polygons for one object

For multiple polymesh objects, the number of polygons is kept consistent with each increment step in the previous test. Multiple 10x10x10 segment boxes, each containing 1,200 polygons, are used to form the polygon count for each test run. The vertex count was greater in the multiple polymesh objects test than in the single polymesh object test by 2(N-1) vertices, where N is the number of boxes. The increase in hload is relatively linear as objects are added to the scene, Figure 7. Further testing gave the GLB and LUB to be 35 objects (42k polygons) and 49 objects (58.8k polygons) for these polymesh objects with non-overlapping polygons. The limit of 1ms was exceeded for 49 objects, therefore not allowing for the true value to be recorded. However, we did notice that with only 35 objects, the

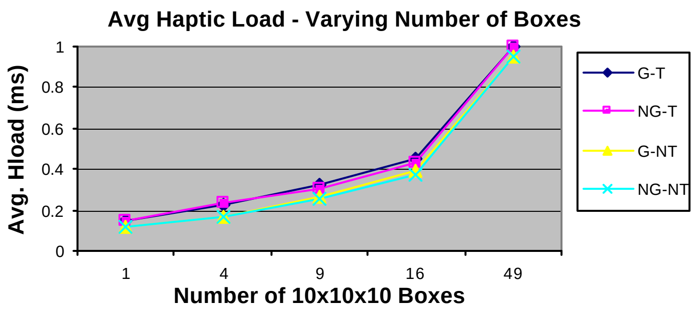

Figure 7: Avg test results for increasing number of objects

number of polygons from the single polymesh object test was reduced from 120k to 42k in the multiple polymesh objects test. Even with an additional 78k polygons the maximum hload was about 80% less.

We estimate that on the average, collision detection for the polymesh objects is about 71-77%, while graphics is about 2-8%, and touching the object is about 19% of the hload. Tests also showed that the graphics increased hload dependent of the number of polygons. Touching the polmesh, however, raised hload on average by about 26%.

4. Conclusion

In this paper, we addressed scene complexity issues based on the maximum number of primitive objects as well as the maximum number of polygons for single and multiple polymesh objects that allow stable haptic interactions when using GHOST. We discussed the idea of specifying a scene complexity stability range. This range was defined by the greatest lower bound (GLB) and the least upper bound (LUB). GLB is the highest scene complexity for which the application always runs with stability and produces no errors. The LUB is a point at which the haptic application will no longer run and produces errors instantly after haptics is initialized. When the scene complexity is between GLB and LUB the application runs, but is unstable and prone to errors. We were also able to estimate the percentage of hload used for the collision detection, graphics, and user touching an object.

Though these initial tests gave us complexity bounds for some haptic applications, more tests need to be performed in order to estimate the feasibility and performance expectations for a general haptic environment that involves many other levels of complexity. For example, other tests results indicate that hload increases if the point of contact is within multiple bounding boxes and increases even more if the point is touching multiple objects. Tests also show that there can be different hloads within a single polymesh object, depending on its topology (eg. corner points, overlapping polygons, etc.). Understanding scene complexity limits plays a key role in creation of haptic applications by providing critical data needed to estimate the feasibility and performance for generic haptic environments.

5. References

[1] Ho, C., Basdogan, C., Srinivasan, M.A., 1999, "An Efficient Haptic Rendering Technique for Displaying 3D Polyhedral Objects and Their Surface Details in Virtual Environments", October 1999 Vol. 8, No. 5, pp. 477-491, Presence: Teleoperators and Virtual Environments.

[2] Acosta, Eric J., Haptic Virtual Environment, M.S. Thesis, Computer Science Department, Texas Tech University, May 2001.

[3] Farida Vahora, Bharti Temkin, Thomas M. Krummel, Paul J. Gorman, “Development of Real -Time Virtual Reality Haptic Application: Real-Time Issues”, *12th IEEE Symposium on Computer-Based Medical Systems* - CBMS 1999, June 18-20, pages 290-295, ISBN 0–7695–0234–2

[4] GHOST Software Developers Toolkit Programmers Guide, SensAble Technologies, Inc.